JOINS Problems

1. Customer Order Summary

Problem: Find all customers and their total number of orders. Include customers who haven't placed any orders yet.

Hint: Use a **LEFT JOIN** between `customers` and `orders`.

---

2. High-Spending Customers

Problem: Identify customers who have spent more than $5000 in total. Include their customer details and total spending.

Hint: Use an **INNER JOIN** between `customers` and `orders`. Aggregate data using `SUM(order_amount)`.

---

3. Orders Without Payments

Problem: List all orders that have not received a payment yet. Include the order ID, customer ID, and payment status.

Hint: Use a **LEFT JOIN** between `orders` and `payments` and filter where the `payment_id` is NULL.

---

4. Product Sales by Category

Problem: Calculate the total sales for each product category. Include categories even if no products have been sold.

Hint: Use a **LEFT JOIN** between `products` and `orders`.

---

5. Duplicate Orders

Problem: Find orders where customers have purchased the same product more than once. Include the customer ID, product ID, and the count of such duplicate orders.

Hint: Use a **JOIN** and a **GROUP BY** with a `HAVING` clause.

---

6. Cross-Selling Analysis

Problem: Identify customers who purchased product A but never purchased product B. Include their details.

Hint: Use a **SELF JOIN** on the `orders` table, comparing customers' product purchases.

---

7. Employee Sales Contribution

Problem: Find the total revenue generated by each employee (sales representative). Include employees who have not made any sales.

Hint: Use a **LEFT JOIN** between `employees` and `orders`.

---

8. Top-Selling Products

Problem: List the top 5 products by total sales (quantity sold and revenue generated). Include their category.

Hint: Use an **INNER JOIN** between `products` and `orders`, aggregate the data, and sort it.

---

10. Refund Analysis

Problem: Identify all refunded orders, their customers, and the refunded amount. Also, include orders that weren't refunded (with refund amount as 0).

Hint: Use a **LEFT JOIN** between `orders` and `refunds`, and use `COALESCE` for refund amounts.

Example Table Structure

If needed, these tables can be used for the problems:

- `customers` (`customer_id`, `customer_name`, `state`)
- `orders` (`order_id`, `customer_id`, `order_date`, `order_amount`, `product_id`, `employee_id`)
- `payments` (`payment_id`, `order_id`, `payment_date`, `payment_amount`)
- `products` (`product_id`, `product_name`, `category_id`, `price`)
- `categories` (`category_id`, `category_name`)
- `employees` (`employee_id`, `employee_name`)
- `regions` (`region_id`, `region_name`)
- `refunds` (`refund_id`, `order_id`, `refund_amount`)